PCS 2190 - DRAW: Repetição de desenhos

Ricardo Nakamura e Romero Tori

2015

1 Introdução

Os programas que criamos até agora têm como característica comum o fato de que o computador executa os comandos uma única vez. O problema é que a criação de padrões mais elaborados se torna muito trabalhosa, podendo precisar de dezenas ou centenas de linhas de programação.

Nesta apostila, veremos que é possível instruir ao Processing para que ele repita um conjunto de comandos. Com isso poderemos criar desenhos complexos de forma mais eficiente. Poderemos também criar animações.

2 Os blocos setup e draw

Para instruir ao Processing que queremos que alguns comandos sejam executados somente uma vez e outros comandos sejam repetidos várias vezes, precisamos dividir nossos programas em dois blocos, como mostrado na listagem a seguir.

```
void setup( ) {
// comandos que são executados só uma vez
// quando o programa começa
}

void draw( ) {
// comandos que são executados repetidamente
// pelo Processing, até fecharmos o programa
}
```

Note que cada um dos blocos possui um nome (**setup** e **draw**) e é delimitado por chaves. Só é possível escrever comandos dentro desses blocos[1].

Os comandos que estiverem dentro do bloco **setup** são executados uma única vez. Os comandos contidos no bloco **draw** são executados repetidamente.

Poderíamos converter qualquer um dos programas das apostilas anteriores simplesmente colocando todos os seus comandos dentro do bloco **setup**, como ilustrado pela listagem a seguir, que mostra nosso primeiro programa em Processing, utilizando a estrutura de blocos.

```
void setup() {
  size(400, 400);
  line(0, 0, 200, 200);
}

void draw() {
}
```

Atividade 1

Modifique o programa anterior, movendo os comandos para o bloco **draw** e observe se ocorre alguma diferença no resultado do programa. O que ocorre se você só mover o comando **line** para o bloco draw?

3 Utilizando variáveis com setup e draw

Ao usarmos variáveis em programas divididos nos blocos setup e draw, elas podem ser criadas dentro ou fora dos blocos. A diferença é que, se uma variável é criada dentro do bloco setup, ela só "existe" nele. O mesmo vale para o bloco draw. Por este motivo, em geral criaremos as variáveis fora dos blocos, como mostrado na listagem a seguir.

[1]Como veremos futuramente, esses "blocos" são funções, que ajudam a organizar programas mais complexos

```
// criamos a variável fora dos blocos
int largura;

void setup() {
  size(400, 400);
  // associamos um valor no bloco setup...
  largura = width/3;
}

void draw() {
  // ... e usamos o valor no bloco draw!
  line(0, 0, largura, 200);
}
```

4 Repetindo comandos

Sabendo que os comandos dentro do bloco **draw** são repetidos automaticamente, podemos utilizar esse bloco para criar desenhos mais complexos. O programa a seguir cria um padrão listrado sem precisar escrever vários comandos **rect**.

```
// criamos a variável fora dos blocos
int posicaoY;

void setup() {
  size(400, 400);
  posicaoY = 0;
  background(0);
  fill(255);
}

void draw() {
  rect(0, posicaoY, width, 10);
  posicaoY = posicaoY + 20;
}
```

Ao executar o programa, é possível ver as listras horizontais sendo desenhadas progressivamente na tela. Isso corresponde à repetição dos comandos dentro do bloco draw. As cinco primeiras vezes que o bloco draw é repetido são simuladas a seguir:

1. desenha um retângulo na posição (0, 0) e incrementa **posicaoY** para 20;

2. desenha um retângulo na posição (0, 20) e incrementa **posicaoY** para 40;
3. desenha um retângulo na posição (0, 40) e incrementa **posicaoY** para 60;
4. desenha um retângulo na posição (0, 60) e incrementa **posicaoY** para 80;
5. desenha um retângulo na posição (0, 80) e incrementa **posicaoY** para 100;

Veja que, sem utilizar a repetição de comandos, seria preciso escrever vinte comandos **rect** pra preencher toda a janela com listras.

Observe também que para aproveitar a repetição dos comandos, é preciso utilizar uma ou mais variáveis cujos valores são modificados a cada repetição. Caso contrário, teremos um único desenho se repetindo continuamente.

Atividade 2

Modifique o comando **rect** do programa anterior para rect(0, 0, 200, 200); e observe o resultado.

Atividade 3

Modifique o programa anterior para desenhar listras verticais.

5 Criando animações

Experimente modificar o programa anterior, das listras verticais, para a listagem seguinte, que basicamente acrescenta um comando **background** dentro do bloco **draw** e observe o resultado:

```
int posicaoY;

void setup() {
  size(400, 400);
  posicaoY = 0;
  fill(255);
}

void draw() {
  background(0);
  rect(0, posicaoY, width, 10);
  posicaoY = posicaoY + 1;
}
```

Devido ao comando **background**, a cada repetição do bloco draw a janela é apagada antes do novo retângulo ser desenhado. Como resultado, temos uma animação bastante simples. Animações computadorizadas em geral funcionam a partir deste princípio: apagar a tela anterior e desenhar uma nova imagem, gerando com isso a ilusão de movimento.

Atividade 4

Crie um programa que faz uma animação de um retângulo que se move horizontalmente.

Atividade 5

Modifique o programa anterior para fazer a barra se mover da direita para a esquerda.

Atividade 6

Modifique o programa anterior para fazer a barra se mover dez vezes mais rápido.

6 Interagindo com o mouse

Na apostila anterior, vimos que o Processing define algumas variáveis automáticas, como **width** e **height**, para representar algumas informações do computador para uso em nossos programas.

Outras duas variáveis automáticas muito úteis são **mouseX** e **mouseY**, que representam as coordenadas, em pixels, do ponteiro do mouse dentro da janela gráfica de nosso programa. Com o uso dessas variáveis e a repetição do bloco **draw**, podemos criar um programa de desenho bem simples:

```
void setup() {
  size(500, 500);
  background(0);
  noStroke();
  fill(255, 0, 0);
}

void draw() {
  ellipse(mouseX, mouseY, 3, 3);
}
```

Note que, como o comando **ellipse** dentro do bloco **draw** é repetido continuamente, a cada instante desenhamos na tela um novo círculo na posição atual do ponteiro do mouse, criando assim uma "caneta virtual".

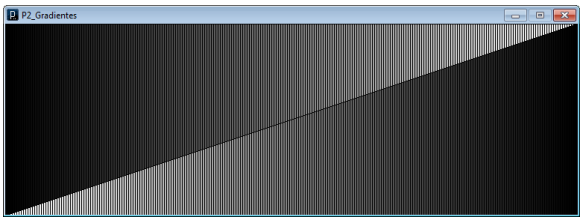
Atividade 7

Modifique o programa anterior para desenhar linhas que conectam a última posição do mouse com a posição atual.

7 Usando repetição para desenhos elaborados

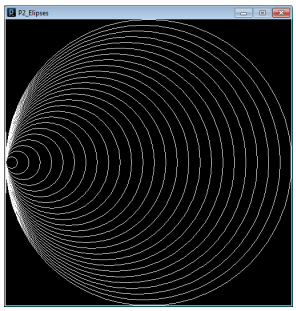
A listagem a seguir ilustra o uso da repetição de comandos para criar um desenho mais complexo. Note que, sem usar a repetição, seria preciso escrever mais de 500 comandos **rect** e **fill**!

```
int posicaoX;
int altura;
int cinza;

void setup() {
  size(768, 255);
  posicaoX = 0;
  altura = height;
  cinza = 0;
  background(0);
}

void draw() {
  fill(cinza);
  rect(posicaoX, 0, 3, altura);
  fill(255 - cinza);
  rect(posicaoX, altura, 3, height - altura);
  // a cada repetição, muda a posição horizontal para a direita
  // diminui o tamanho do retângulo superior e aumenta a
  // intensidade do tom de cinza.
  posicaoX = posicaoX + 3;
  altura = altura - 1;
  cinza = cinza + 1;
}
```



Atividade 8

Faça um programa para reproduzir o desenho abaixo, usando o bloco **draw** para repetição.



Atividade 9

Faça um programa para reproduzir o desenho abaixo, usando o bloco **draw** para repetição.